

An ADMM algorithm for solving a proximal bound-constrained quadratic program

Miguel Á. Carreira-Perpiñán

Electrical Engineering and Computer Science, University of California, Merced
<http://eecs.ucmerced.edu>

December 29, 2014

Abstract

We consider a proximal operator given by a quadratic function subject to bound constraints and give an optimization algorithm using the alternating direction method of multipliers (ADMM). The algorithm is particularly efficient to solve a collection of proximal operators that share the same quadratic form, or if the quadratic program is the relaxation of a binary quadratic problem.

1 Introduction

We consider the convex *proximal bound-constrained quadratic program* (QP):

$$\min_{\mathbf{x} \in \mathbb{R}^D} f(\mathbf{x}) + \frac{\mu}{2} \|\mathbf{x} - \mathbf{v}\|^2 \text{ s.t. } \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \quad f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} \quad (1)$$

where $\mu > 0$, $\mathbf{v}, \mathbf{l}, \mathbf{u}, \mathbf{b} \in \mathbb{R}^D$ and the $D \times D$ matrix \mathbf{A} is symmetric positive definite or semidefinite. The problem has the form of a convex proximal operator (Combettes and Pesquet, 2011; Moreau, 1962; Rockafellar, 1976) $\mathbf{x} = \text{prox}_{f/\mu}(\mathbf{v})$ (with $\text{dom}(f) = [\mathbf{l}, \mathbf{u}]$), and a unique solution. We will also be interested in solving a collection of such QPs that all have the same matrix \mathbf{A} but different vectors $\mathbf{b}, \mathbf{v}, \mathbf{l}$ and \mathbf{u} .

It is possible to solve problem (1) in different ways, but we want to take advantage of the structure of our problem, namely the existence of the strongly convex μ term, and the fact that the N problems have the same matrix \mathbf{A} . We describe here one algorithm that is very simple, has guaranteed convergence without line searches, and takes advantage of the structure of the problem. It is based on the alternating direction method of multipliers (ADMM), combined with a direct linear solver and caching the Cholesky factorization of \mathbf{A} .

Motivation Problem (1) arises within a step in the binary hashing algorithm of Carreira-Perpiñán and Raziperchikolaei (2015). The step involves N independent subproblems

$$\min_{\mathbf{x}_n \in \{0,1\}^D} \frac{1}{2} \|\mathbf{C} \mathbf{x}_n - \mathbf{d}_n\|^2 + \frac{\mu}{2} \|\mathbf{x}_n - \mathbf{v}_n\|^2 \quad n = 1, \dots, N.$$

Intuitively, subproblem n tries to map linearly a binary vector \mathbf{v}_n onto a real vector \mathbf{d}_n with minimal error, given a mapping of matrix \mathbf{C} . Hence, each subproblem is of the form of (1), but where f is a least-squares function (where the rectangular matrix \mathbf{C} is common to all subproblems) and the variables are binary. Relaxing these subproblems results in N proximal bound-constrained QPs of the form of (1).

2 Solving a QP using ADMM

We briefly review how to solve a quadratic program (QP) using the alternating direction method of multipliers (ADMM), following Boyd et al. (2011). Consider the QP

$$\min_{\mathbf{x}} \quad \frac{1}{2} \mathbf{x}^T \mathbf{P} \mathbf{x} + \mathbf{q}^T \mathbf{x} \quad (2)$$

$$\text{s.t.} \quad \mathbf{A} \mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} \quad (3)$$

over $\mathbf{x} \in \mathbb{R}^D$, where \mathbf{P} is positive (semi)definite. To use ADMM, we first introduce new variables $\mathbf{z} \in \mathbb{R}^D$ so that we replace the inequalities with an indicator function $g(\mathbf{z})$, which is zero in the nonnegative orthant $\mathbf{z} \geq \mathbf{0}$ and ∞ otherwise. Then we write the problem as

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) + g(\mathbf{z}) \quad (4)$$

$$\text{s.t.} \quad \mathbf{x} = \mathbf{z} \quad (5)$$

where

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{P} \mathbf{x} + \mathbf{q}^T \mathbf{x}, \quad \text{dom}(f) = \{\mathbf{x} \in \mathbb{R}^D: \mathbf{A} \mathbf{x} = \mathbf{b}\}$$

is the original objective with its domain restricted to the equality constraint. The augmented Lagrangian is

$$\mathcal{L}(\mathbf{x}, \mathbf{z}, \mathbf{y}; \rho) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T (\mathbf{x} - \mathbf{z}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z}\|^2 \quad (6)$$

and the ADMM iteration has the form:

$$\mathbf{x} \leftarrow \arg \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{z}, \mathbf{y}; \rho)$$

$$\mathbf{z} \leftarrow \arg \min_{\mathbf{z}} \mathcal{L}(\mathbf{x}, \mathbf{z}, \mathbf{y}; \rho)$$

$$\mathbf{y} \leftarrow \mathbf{y} + \rho(\mathbf{x} - \mathbf{z})$$

where \mathbf{y} is the dual variable (the Lagrange multiplier estimates for the constraint $\mathbf{x} = \mathbf{z}$), and the updates are applied in order and modify the variables immediately. Here, we use the scaled form of the ADMM iteration, which is simpler. It is obtained by combining the linear and quadratic terms in $\mathbf{x} - \mathbf{z}$ and using a scaled dual variable $\boldsymbol{\zeta} = \mathbf{y}/\rho$:

$$\mathbf{x} \leftarrow \arg \min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z} + \boldsymbol{\zeta}\|^2 \right)$$

$$\mathbf{z} \leftarrow \arg \min_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z} + \boldsymbol{\zeta}\|^2 \right)$$

$$\boldsymbol{\zeta} \leftarrow \boldsymbol{\zeta} + \mathbf{x} - \mathbf{z}.$$

Since $g(\mathbf{x})$ is the indicator function for the nonnegative orthant, the solution of the \mathbf{z} -update is simply to threshold each entry in $\mathbf{x} + \boldsymbol{\zeta}$ by taking its nonnegative part. Finally, the ADMM iteration is:

$$\mathbf{x} \leftarrow \arg \min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z} + \boldsymbol{\zeta}\|^2 \right) \quad (7a)$$

$$\mathbf{z} \leftarrow (\mathbf{x} + \boldsymbol{\zeta})_+ \quad (7b)$$

$$\boldsymbol{\zeta} \leftarrow \boldsymbol{\zeta} + \mathbf{x} - \mathbf{z} \quad (7c)$$

where the updates are applied in order and modify the variables immediately, $(t)_+ = \max(t, 0)$ applies elementwise, and $\|\cdot\|$ is the Euclidean norm. The penalty parameter $\rho > 0$ is set by the user, and $\mathbf{z} = (z_1, \dots, z_D)^T$ are the Lagrange multiplier estimates for the inequalities. The \mathbf{x} -update is a quadratic objective whose solution is given by a linear system. The ADMM iteration consists of very simple updates to the relevant variables, but its success crucially relies in being able to solve the \mathbf{x} -update efficiently. Convergence of the ADMM iteration (7) to the global minimum of problem (2) in value and to a feasible point is guaranteed for any $\rho > 0$.

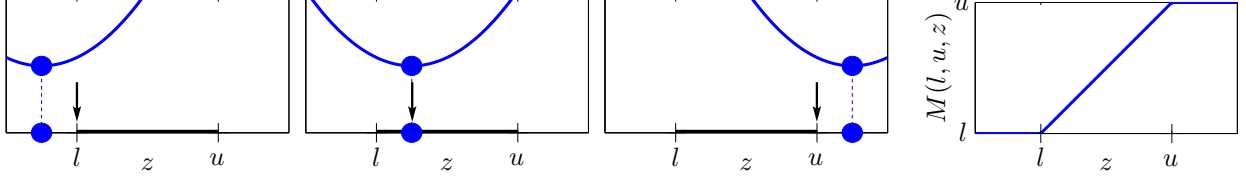


Figure 1: *Left*: three possible cases for the location of the solution for the scalar box-constrained QP in the \mathbf{z} step. *Right*: the solution $M(l, u, z)$ is the median of l , u and z .

3 Application to our QP

In order to apply ADMM to our QP (1), we make the identification $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x}$, defined over the entire \mathbb{R}^D , and $g(\mathbf{z}) = \frac{\mu}{2}\|\mathbf{z} - \mathbf{v}\|^2$, defined for $\mathbf{l} \leq \mathbf{z} \leq \mathbf{u}$, and want to minimize $f(\mathbf{x}) + g(\mathbf{z})$ s.t. $\mathbf{x} = \mathbf{z}$. The augmented Lagrangian is

$$\mathcal{L}(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}; \rho) = \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} + \frac{\mu}{2}\|\mathbf{z} - \mathbf{v}\|^2 + \boldsymbol{\lambda}^T(\mathbf{x} - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}\|^2 \text{ s.t. } \mathbf{l} \leq \mathbf{z} \leq \mathbf{u}$$

and the ADMM iteration is:

$$\begin{aligned} \mathbf{x} &\leftarrow \arg \min_{\mathbf{x}} \left(\frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z} + \boldsymbol{\zeta}\|^2 \right) \\ \mathbf{z} &\leftarrow \arg \min_{\mathbf{z}} \left(\frac{\mu}{2}\|\mathbf{z} - \mathbf{v}\|^2 + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z} + \boldsymbol{\zeta}\|^2 \right) \text{ s.t. } \mathbf{l} \leq \mathbf{z} \leq \mathbf{u} \\ \boldsymbol{\zeta} &\leftarrow \boldsymbol{\zeta} + \mathbf{x} - \mathbf{z}. \end{aligned}$$

We now solve the steps over \mathbf{x} and \mathbf{z} . The step over \mathbf{x} is an unconstrained strongly convex quadratic function whose unique minimizer is given by a linear system. The step over \mathbf{z} separates over each component z_1, \dots, z_D of \mathbf{z} because both the objective and the constraints are separable. For each component z_d , $d = 1, \dots, D$, we have to minimize an upwards parabola defined in $[l_d, u_d]$, whose solution is the median $M(l_d, u_d, z_d^*)$ of l_d , u_d and the parabola vertex $z_d^* = \frac{\mu v_d + \rho(x_d + \zeta_d)}{\mu + \rho}$ (see fig. 1). Finally, we obtain the following updates, which are iterated in order until convergence:

$$\mathbf{x} \leftarrow (\mathbf{A} + \rho \mathbf{I})^{-1}(\mathbf{b} + \rho(\mathbf{z} - \boldsymbol{\zeta})) \quad (8a)$$

$$\mathbf{z} \leftarrow M\left(\mathbf{l}, \mathbf{u}, \frac{\mu \mathbf{v} + \rho(\mathbf{x} + \boldsymbol{\zeta})}{\mu + \rho}\right) \quad (8b)$$

$$\boldsymbol{\zeta} \leftarrow \boldsymbol{\zeta} + \mathbf{x} - \mathbf{z} \quad (8c)$$

where the median applies elementwise to its vector arguments, \mathbf{x} are the primal variables, \mathbf{z} the auxiliary (consensus) variables, and $\boldsymbol{\zeta}$ the scaled Lagrange multiplier estimates for \mathbf{z} . Moving the inequalities into the \mathbf{z} subproblem results in a very simple step simply involving elementwise thresholding operations.

The iteration (8) has a number of advantages. It is very simple to implement. It requires no line searches and has only one user parameter, the penalty parameter ρ , for which a good default value can be computed (see below). The algorithm converges for any positive value of ρ . The steps are fast and we can stop at any time with a feasible iterate. The algorithm is particularly convenient if we have to solve N proximal operators of the form (1) that have the same matrix \mathbf{A} , since the Cholesky factor is computed just once and used by all operators. Finally, if the QP is the relaxation of a binary problem (i.e., \mathbf{x} was originally in $\{0, 1\}^D$ but was relaxed to $[0, 1]^D$), we need not converge to high accuracy because the final result will be binarized a posteriori.

Computational complexity Each step in (8) is $\mathcal{O}(D)$ except for the \mathbf{x} step, which is the most costly because of the linear system. This may be solved efficiently in one of the two following ways:

- Preferably, by caching the Cholesky factorization of $\mathbf{A} + \rho \mathbf{I}$ (i.e., $\mathbf{R}\mathbf{R}^T = \mathbf{A} + \rho \mathbf{I}$ where \mathbf{R} is upper triangular). If \mathbf{A} is dense, computing the Cholesky factor is a one-time cost of $\mathcal{O}(\frac{1}{3}D^3)$, and solving

the linear system is $\mathcal{O}(D^2)$ by solving two triangular systems. If \mathbf{A} is large and (sufficiently) sparse, computing the Cholesky factor and solving the linear system are both possibly $\mathcal{O}(D)$. One should use a good permutation to reduce fill-in. This is feasible with relatively small dimensions D if using a dense \mathbf{A} , or with large dimensions as long as \mathbf{A} is sufficiently sparse that the Cholesky factorization does not add so much fill that it can be stored.

- By using an iterative linear solver such as conjugate gradients (Nocedal and Wright, 2006), initialized with a warm start, preconditioned, and exiting it before convergence, so as to carry out faster, inexact \mathbf{x} -updates. This is better for large problems where the Cholesky factorization adds too much fill.

Thus, each iteration of the algorithm is cheap. In practice, for good values of ρ , the algorithm quickly approaches the solution in the first iterations and then converges slowly, as is known with ADMM algorithms in general. However, since each iteration is so cheap, we can run a large number of them if high accuracy is needed. As a sample runtime, for a collection of $N = 60\,000$ problems in $D = 32$ variables having the same, dense matrix \mathbf{A} , solving all N problems to high accuracy takes around 1 s in a PC.

Initialization If the QP problem (1) is itself a subproblem in a larger problem, one should warm-start the iteration of eq. (8) from the values of \mathbf{z} and $\boldsymbol{\zeta}$ in the previous outer-loop iteration. Otherwise, we can simply initialize $\mathbf{z} = \mathbf{v}$ and $\boldsymbol{\zeta} = \mathbf{0}$.

Stopping criterion We stop when the relative change in \mathbf{z} is less than a set tolerance (e.g. 10^{-5}). We return as approximate solution the value of \mathbf{z} , which is feasible by construction, while \mathbf{x} need not be (upon convergence, $\mathbf{x} = \mathbf{z}$).

Optimal penalty parameter ρ The speed at which ADMM converges depends significantly on the quadratic penalty parameter ρ (Boyd et al., 2011). Little work exists on how to select ρ so as to achieve fastest convergence. Recently, for QPs, Ghadimi et al. (2013) suggest to use $\rho^* = \sqrt{\sigma_{\min}\sigma_{\max}}$ where σ_{\min} and σ_{\max} are the smallest (nonzero) and largest eigenvalue of the matrix \mathbf{A} .

Matlab code A Matlab implementation of the algorithm is available from the author. The following Matlab code implements a basic form of the algorithm, using the Cholesky factor in the \mathbf{x} -update linear system, and assuming the bounds $[l, u]$ are the same for each of the N quadratic programs. It applies the algorithm synchronously to all the QPs, which means each QP runs the same number of iterations. This is inefficient, because the runtime is driven by the slowest variable to converge, but the code is better vectorized in Matlab. An implementation in C should handle each QP separately, particularly in a parallel or distributed implementation.

```
function [Z,Y,X] = proxbqp(V,m,A,B,l,u,Z,Y,r,maxit,tol)

[D,N] = size(V); R = chol(A+r*eye(D,D)); Rt = R'; Zold = zeros(D,N);
for i=1:maxit
    X = R \ (Rt \ (B + r*(Z-Y)));
    Z = (m*V+r*(X+Y))/(m+r); Z(Z<l) = l; Z(Z>u) = u;
    Y = Y + X - Z;
    if max(abs(Z(:)-Zold(:))) < tol break; end; Zold = Z;
end
```

References

- S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1): 1–122, 2011.
- M. Á. Carreira-Perpiñán and R. Raziperchikolaei. Hashing with binary autoencoders. Unpublished manuscript, Jan. 2015.

- P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer Series in Optimization and Its Applications, pages 185–212. Springer-Verlag, 2011.
- E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems. arXiv:1306.2454 [math.OC], July 3 2013.
- J.-J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *C. R. Acad. Sci. Paris Sér. A Math.*, 255:2897–2899, 1962.
- J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, second edition, 2006.
- R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control and Optim.*, 14(5):877–898, 1976.